ADDING A 24C04 SERIAL EEPROM TO YOUR BOARD

A. Read the following data sheet and application notes to gain an understanding of the 24C04 and I^2C protocol:

1. Device Data Sheet – NM24C04/05
   http://www.fairchildsemi.com/ds/NM/NM24C04.pdf

   Comments: Read this to learn about the IIC (I^2C) interface protocol and the 24C04 serial EEPROM device that you will be adding to your board. If you feel like you would like some more background on I^2C and EEPROM basics, you may want to also read this application note:
   http://www.microchip.com/Download/Appnote/Category/EEPROMS/00536.pdf
   Read the introduction sections and the '2-WIRE BUS OPERATION PRIMER'. Skip the 3-wire bus operation primer.

2. AN537 from Microchip
   http://www.microchip.com/Download/Appnote/Category/EEPROMS/00537.pdf

   Comments: Read this to learn about how EEPROM endurance figures in to some industry design examples. Also, get a little intuition about how programming works at the device level.

   Time: This should be a **quick** read. It is interesting to understand how endurance must be considered in products developed by industry. The device level programming of an EEPROM memory bit is also pretty fascinating.

3. Optional Application Notes Reading
   http://www.fairchildsemi.com/an/AN/AN-794.pdf
   http://www.fairchildsemi.com/an/AN/AN-822.pdf

   Comments: If you are still craving more information, read these app notes as well. However, they do **not** provide too much new information for the purposes of this class.

B. Add the EEPROM hardware to your board.
Comments: From a hardware standpoint, adding the EEPROM to your board is relatively simple. There are very few connections to be made and the wealth of information in the application notes and data sheets is helpful.

- Remember to use pull up resistors on SCL and SDA!
- If you want to leverage the 8051 code in AN614 (URL below in section C) be sure to read the PDF file on how to connect your hardware to match the code. **Note: the schematic shown in AN614 is missing a pull up resistor!
- Write protect is not needed.
- Be sure to connect the address pins properly. For now you will only be adding one serial EEPROM to your board.

C. Leveraging 8051 assembly code from the WWW.

Comments: Microchip has a great application note on interfacing a serial EEPROM with the 8051. The code that can be downloaded with the application note has been tested with success. After some minor changes, you will be able to assemble it. First, read through the code to gain an understanding of the following functions: BYTEW, OUTS, OUT, STOP, PAGEW, BLKRD, BYTERD, IN, ACKTST, CREAD. Second, you may want to verify that the code creates a 100kHz clock on SCL. Third, verify that the correct I^{2}C protocol is being used by the functions. Debugging and testing will be much easier if the code makes complete sense.

The application note is located at:
http://www.microchip.com/Download/Appnote/Category/EEPROMS/00614b.pdf

The application note source code (in a ZIP file) can be downloaded from this web page:
http://www.microchip.com/10/Appnote/Category/EEPROMS/00614/indexZIP1.htm

Code Changes: (Line numbers are given in the application note)
Since we are using a different assembler than what was used by the engineer that wrote the AN614 code, some modifications must be made before the source code can be assembled.

1. Delete the $MOD51 on line #1.
2. Delete the DSEG on line #34
3. Delete the CSEG on line #63
4. There is an ORG on line #35 and on line #64. Change or delete these to meet your needs. If you do not delete the ORGs, you will need to insert a space or TAB before each ORG in order to assemble the code.
5. Using an editor, replace all CALL instructions with ACALL
6. Using an editor, replace all JMP instructions with AJMP

After making these code changes, you should be able to assemble your code.

D. Testing/Debugging Tips

✓ To start, you might want to write a simple menu that will allow you to call the AN614 functions via the MON51 interface. However, if you do this, you may have to modify the AN614 functions so that they get their operands out of memory instead of registers. If you just use MON51 to change register values as a means of passing arguments, the data may get corrupted before your serial EEPROM function gets called.

✓ Using state analysis on the logic analyzers in the lab can be a great way to test that the proper data is being transmitted between the 8051 and the 24C04. Simply attach the analyzer CLK probe to SCL and a data probe to SDA. Set the analyzer clock to trigger on a positive SCL edge so that you latch in valid data. Note, however, that you will not be able to observe the START bit with this scheme. Since there is no rising SCL clock edge for the START bit (see the timing diagrams in the 24C04 data sheet), the analyzer will not latch the bit in.